

HomeWork - 2

Mustafa Tokat

December 5, 2020

1 Introduction

First, I would like to thank you very much again for giving me such a great opportunity. I am looking forward to the lessons and the many new things I can learn. I must confess that I take pleasure in learning this subject. At the same time I am so sorry because my English is not sufficient for writing knowledge rules. This is my first time writing something of this length. Indeed I am not used to this situation but I aim to explain myself the best I can.

Furthermore, I preferred to add to the end of this document to all coding documents. I thought you would be distracted while you read if I had done otherwise. Therefore you will find the aforementioned documents below.

2 Analyze of Problems

Problem 1

This polynomial is reducible: $p(x) = x^5 + x^4 + 1$ Discover the period(s) of the sequence produced by the LFSR using it as its connection polynomial

There is a reducible polynomial my hands. This is very important in terms of two. At first, this polynomial will not produce maximal period. We know to 2 theorem discovered by Solomon Golomb. The first of these, if we want to produce an n -bit LFSR, of this polynomial have to have maximal period. If period of this polynomial is not maximal, it can produce each time different sequences and this is the case 2 different DRNG machines can not run compatible.

Then we should remember theorems of Solomon Golomb's:

Golomb's first theorem. *If the connection polynomial (of degree n) of an n -bit LFSR is reducible, then its period is not maximal ($\neq 2^n - 1$)*

And again let's remember of:

Golomb's second theorem. *If the connection polynomial of degree n is a primitive polynomial, then the associated LFSR is maximal, with period $2^n - 1$.*

Then we can write: $x^5 + x^4 + 1 == (c_4, c_3, c_2, c_1, c_0) == (1, 0, 0, 0, 1)$ Initial state is important because of this polynomial is not irreducible and primitive polynomial. We will see right now.

Table 1: Initial state for 0, 0, 0, 0, 1, periods of number 23

$(c_4, c_3, c_2, c_1, c_0)$	1	0	0	0	1
$(s_4, s_3, s_2, s_1, s_0)$	1	1	1	0	0
					initial value
	1	1	1	1	0
	1	1	1	1	1
	0	1	1	1	1
	1	0	1	1	1
	0	1	0	1	1
	1	1	0	0	1
	0	1	0	0	1
	1	0	1	0	1
	0	1	0	1	0
	0	0	1	0	1
	1	0	0	1	0
	1	1	0	0	1
	0	1	1	0	0
	0	0	1	1	0
	0	0	0	1	1
	1	0	0	0	1
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1
	1	0	0	0	0
	1	1	0	0	0
	1	1	1	0	0
					Repeat

Table 4: Initial state for 1, 1, 1, 0, 1, periods of number 7

$(c_4, c_3, c_2, c_1, c_0)$	1	0	0	0	1	
$(s_4, s_3, s_2, s_1, s_0)$	1	1	1	0	1	initial value
	0	1	1	1	0	
	0	0	1	1	1	
	1	0	0	1	1	
	0	1	0	0	1	
	1	0	1	0	0	
	1	1	0	1	0	
	1	1	1	0	1	repeat

Result: I tried 4 different initial state values. As it seen from tables, 4 different initial state values produced 4 different periods. Also, if given polynomial were primitive polynomial whatsoever initial state value would produce the same periods, in unique period number $2^5 - 1 = 31$.

Problem 2

Prove that $p(x) = x^{10} + x^3 + 1$ is primitive over $GF(2)$.

A polynomial have to be irreducible for primitive polynomial. We can understand that; at first, given polynomial is divided with $x^e + 1$. After, n value continues to increment 1 by e each time, until divided without remainder. However, (e) value can be divided with a few number as without remainder. But we want the first value to satisfy $e = 2^n - 1$. We can do it that:

starting from n;

$$1. \text{ trying: } \frac{2^e - 1}{x^{10} + x^3 + 1}$$

$$2. \text{ trying: } \frac{2^{e+1} - 1}{x^{10} + x^3 + 1}$$

$$3. \text{ trying: } \frac{2^{e+2} - 1}{x^{10} + x^3 + 1}$$

This process continues until $e = 2^n - 1$ first value of e.

After briefly explaining the process, I want to present my Python codes.

Listing 1: My code particle which test to whether primitive polynomial

```
import numpy as np
from sympy import prem
from sympy.abc import x

power1 = int(input('Power1:_'))
power2 = int(input('Power2:_'))
range1 = int(input('Range1:_'))
range2 = int(input('Range2:_'))

def primitiveTest():

    for i in range(range1, range2):

        m = ((prem (x**i + 1, x**power1 + x**power2 + 1, modulus = 2) == 0) and
             i == 2**power1 - 1)

        if m == True :

            print(i)

        else:
            i + 1
            print('Not Primitive')

primitiveTest()
```

Result: $x^{10} + x^3 + 1$ is a primitive polynomial according to test
result.(Figure 1)

Problem 3

Consider the following sequence, and construct the smallest LFSR producing this sequence using the Berlekamp-Massey algorithm:
1001101001000010101110110001111100110

Indeed we will use to advantage that LFSR has a linear equation system, of course, according to adversary. So, we need to have two things.

- to r values
- the sequence that I had obtained before (as s_0, s_1, s_n)

We will use a kind of deductive method. We actually will think as matrix system.

$$s_n = c_{n-1}s_{n-1} + c_{n-2}s_{n-2} + \dots + c_1s_1 + c_0s_0$$

$$s_{n+1} = c_{n-1}s_n + c_{n-2}s_{n-1} + \dots + c_1s_2 + c_0s_1$$

$$s_{n+2} = c_{n-1}s_{n+1} + c_{n-2}s_n + \dots + c_1s_3 + c_0s_2$$

.

.

.

$$s_{2n-2} = c_{n-1}s_{2n-3} + c_{2n-4}s_n + \dots + c_1s_{n-1} + c_0s_{n-2}$$

$$s_{2n-1} = c_{n-1}s_{2n-2} + c_{2n-3}s_n + \dots + c_1s_n + c_0s_{n-1}$$

If we think of it as a matrix we will need 's' values of 2 times the 'c' values. We know that 'r' values actually mean of 'c' values. And we will seek 'n' values with exhaustive method. I sought $n = 2$, $n = 3$, $n = 4$, and I could not validate with other 's' values. I did not try $n = 1$ because it would produce all 0 or all 1. I found $n = 5$ bit LFSR construct according to the code I prepared, which I will add end of my homework. Therefore I will give to following example according to $n = 5$.

I need to do that:

$$\begin{bmatrix} s_4 & s_3 & s_2 & s_1 & s_0 \\ s_5 & s_4 & s_3 & s_2 & s_1 \\ s_6 & s_5 & s_4 & s_3 & s_2 \\ s_7 & s_6 & s_5 & s_4 & s_3 \\ s_8 & s_7 & s_6 & s_5 & s_4 \end{bmatrix} * \begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \end{bmatrix}$$

So, I will identify to Python and find to $(c_4, c_3, c_2, c_1, c_0)$:

$$\begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

I need to validation, 'c' values always same, and that:

$$\begin{bmatrix} s_{19} & s_{18} & s_{17} & s_{16} & s_{15} \\ s_{20} & s_{19} & s_{18} & s_{17} & s_{16} \\ s_{21} & s_{20} & s_{19} & s_{18} & s_{17} \\ s_{22} & s_{21} & s_{20} & s_{19} & s_{18} \\ s_{23} & s_{22} & s_{21} & s_{20} & s_{19} \end{bmatrix} * \begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_{20} \\ s_{21} \\ s_{22} \\ s_{23} \\ s_{24} \end{bmatrix}$$

I check it and I see that $(s_{20}, s_{21}, s_{22}, s_{23}, s_{24}) = (1, 0, 1, 1, 0)$

Result: $n = 5$ the smallest LSFR producing according to this sequence. Besides connection polynomial of this sequence: $x^5 + x^3 + 1$ according to $(c_4, c_3, c_2, c_1, c_0) = (0, 1, 0, 0, 1)$ (Figure 2)

P.S.: In order for you not to be disturbed while reading, I will only share the relevant code snippets, you can find all of them in the attachment.

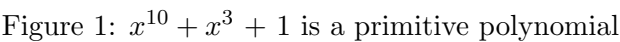


Figure 1: $x^{10} + x^3 + 1$ is a primitive polynomial


```

1  import numpy as np
2  import sympy as sym
3  from sympy import Matrix
4  from sympy import *
5  from sympy import poly
6  from sympy.abc import x
7  init_printing(use_unicode=True)
8
9
10 Run Cell | Run Below | Debug Cell
11 ##### define to variables, my goal is to operate o indices.
12 sValues = (1,0,0,1,1,0,1,0,0,1,0,0,0,1,0,1,1,1,0,1,1,0,0,0,1,1,1,1,0,0,1,1,0)
13
14 n2 = Matrix([
15     [sValues[1], sValues[0]],
16     [sValues[2], sValues[1]]
17 ])
18
19
20 n3 = Matrix([
21     [sValues[2], sValues[1], sValues[0]],
22     [sValues[3], sValues[2], sValues[1]],
23     [sValues[4], sValues[3], sValues[2]]
24 ])
25
26 n4 = Matrix([
27     [sValues[3], sValues[2], sValues[1], sValues[0]],
28     [sValues[4], sValues[3], sValues[2], sValues[1]],
29     [sValues[5], sValues[4], sValues[3], sValues[2]],
30     [sValues[6], sValues[5], sValues[4], sValues[3]]
31 ])
32
33
34 n5 = Matrix([
35     [sValues[4], sValues[3], sValues[2], sValues[1], sValues[0]],
36     [sValues[5], sValues[4], sValues[3], sValues[2], sValues[1]],
37     [sValues[6], sValues[5], sValues[4], sValues[3], sValues[2]],
38     [sValues[7], sValues[6], sValues[5], sValues[4], sValues[3]],
39     [sValues[8], sValues[7], sValues[6], sValues[5], sValues[4]]
40 ])
41
42 ##### for n = 5;
43 def for_n5():
44     cValues_n5 = n5.inv_mod(2) * Matrix([sValues[5], sValues[6], sValues[7], sValues[8], sValues[9]])
45     valid_n5 = Matrix([sValues[19], sValues[18], sValues[17], sValues[16], sValues[15]],
46                        [sValues[20], sValues[19], sValues[18], sValues[17], sValues[16]],
47                        [sValues[21], sValues[20], sValues[19], sValues[18], sValues[17]],
48                        [sValues[22], sValues[21], sValues[20], sValues[19], sValues[18]],
49                        [sValues[23], sValues[22], sValues[21], sValues[20], sValues[19]]
50                        ) % 2 * cValues_n5
51
52     valid_n5 = valid_n5 % 2
53
54     if valid_n5 == Matrix([sValues[20], sValues[21], sValues[22], sValues[23], sValues[24]]):
55         print('({:d} - bit LFSR construct'.format(5))
56         print('Validation for s(20-24): ', valid_n5, '== s20: {:d}, s21: {:d}, s22: {:d}, s23: {:d}, s24: {:d}'.format(sValues[20], sValues[21], sValues[22], sValues[23], sValues[24]))
57     else:
58         print(valid_n5, '!= ({:d}, {:d}, {:d}, {:d}, {:d})'.format(sValues[20], sValues[21], sValues[22], sValues[23], sValues[24]))
59
60     print('Coeffs of Connection Polynomial: ', cValues_n5 % 2)
61
62 Run Cell | Run Above | Debug Cell
63 Run to Functions
64 for_n5()
65
66
67 Microsoft Windows [Version 10.0.18363.1198]
68 (c) 2019 Microsoft Corporation. All rights reserved.
69
70 C:\Users\mstf\Desktop\Cryptogrphy Articles\My Homeworks\Homework-2\My-Homework-2>C:\Users\mstf\anaconda3\Scripts
71 /activate
72
73 (base) C:\Users\mstf\Desktop\Cryptogrphy Articles\My Homeworks\Homework-2\My-Homework-2>conda activate C:\Users\m
74 stf_anaconda3
75
76 (base) C:\Users\mstf\Desktop\Cryptogrphy Articles\My Homeworks\Homework-2\My-Homework-2>C:\Users\mstf_anaconda3/
77 python.exe "c:/Users/mstf/Desktop/Cryptogrphy Articles/My Homeworks/Homework-2/My-Homework-2/Homework-2/find2Conn
78 Poly.py"
79 5 - bit LFSR construct
80 Validation for s(20-24): Matrix([[1], [0], [1], [1], [0]]) == s20: 1, s21: 0, s22: 1, s23: 1, s24: 0
81 Coeffs of Connection Polynomial: Matrix([[0], [1], [0], [0], [1]])
82
83 (base) C:\Users\mstf\Desktop\Cryptogrphy Articles\My Homeworks\Homework-2\My-Homework-2>

```

Figure 2: My codes for 5 bit LFSR construct according to given sequence